Implantation temps réel sur GPU d'un écran d'épingles dynamique

Kevin Sillam, Matthieu Evrard, Annie Luciani

Implantation temps réel sur GPU d'un écran d'épingles dynamique

Kevin Sillam, Matthieu Evrard, Annie Luciani
ICA laboratory – INPG 46 avenue Félix Viallet 38000 Grenoble, France
Mail : prénom.nom@imag.fr

Résumé

Cet article présente une implantation temps réel sur processeur graphique de la méthode dite de l'écran d'épingles dynamique inspiré de la méthode de gravure d'Alexandre Alexeïeff. Cette méthode initialement, proposée par [HL02] et fondée sur un formalisme masses-interactions, était trop coûteuse pour pouvoir s'insérer dans une application interactive. Cet article résout ce frein majeur à l'exploration des possibilités dynamiques de la méthode. Il présente alors un certain nombre d'applications obtenues en temps-réel : (1) la visualisation de modèles particulaires simulés sur une machine amont en temps réel ou en temps différé, (2) l'utilisation de l'écran d'épingles dynamique comme modèle physique pour des scènes 3D complexes, (3) une gravure directe par l'intermédiaire de la souris.

Mots clés: modélisation physique, visualisation de modèles particulaires, implémentation sur GPU, écran d'épingles, phyxel.

1 Introduction

Au cœur du processus de gravure, se trouve une relation entre deux corps physiques, l'un, dur, que l'on peut nommer « marqueur » vient modifier l'état de l'autre, appelé alors « support ».
La roche sédimentaire comme fossile sera un support durable pour rendre compte de la sépulture d'un animal préhistorique. La surface de l'eau au contraire sera un support plus volubile de la mémoire d'un galet qui a ricoché. Quoi qu'il en soit, le support est toujours une mémoire, une trace, du passage d'un marqueur, de sa dynamique, de l'action qu'il a eu sur le support, que cette mémoire soit ou non éphémère.
Un certain nombre de films d'animation ont utilisé la gravure comme procédé central de création des images. La gravure permet en effet à l'artiste d'obtenir des formes aux contours flous, insaisissables.
Parmi la multitude de ces œuvres, on peut retenir l'approche originale d'Alexandre Alexeïeff [AAA96]. Ce graveur de formation met en place au début des années 30, un support de gravure fait d'une multitude d'épingles coulissant perpendiculairement à un plan de fixation. Alexeïeff vient alors graver ce support en déplaçant les épingles à l'aide d'objets divers faisant office de marqueurs. Une source de lumière éclaire cet « écran d'épingles » et fait apparaître des blancs dans les zones où les épingles ont l'altitude la plus élevée et des noirs dans les zones où elles sont les plus enfoncées. La succession des photos, prises de cet écran dans des configurations d'altitudes modifiées au fur et à mesure par le graveur, crée artificiellement le mouvement lors de leur projection séquencée.
Habibi et al. [HL02] partent de l'idée d'Alexeïeff pour résoudre une partie du problème lié à la visualisation de modèles particulaires. Des particules en interaction dont le mouvement est simulé produisent des comportements riches et variés. En revanche, une difficulté majeure inhérente à ce type de formalisme pour la production d'images animées réside dans le fait qu'il est difficile de produire des séquences d'images animées par le rendu direct des masses ponctuelles en mouvement, les masses ponctuelles n'ayant pas de spatialité. D'une manière générale, il est donc nécessaire de développer des méthodes qui étendent la spatialité de ces masses ponctuelles pour compléter la chaîne de production d'images animées par modèle physique particulaire. L'idée d'Habibi et al. est

de venir marquer un écran d'épingles simulé lui-même par modèle physique masses-interactions par les masses en mouvement d'un modèle physique amont. L'écran d'épingles dynamique d'Habibi et al. a donc cela de différent de celui d'Alexeïeff que l'ordinateur permet de générer un mouvement pour les épingles, leur conférant un comportement dynamique.
Les auteurs obtiennent des images réellement convaincantes de fumées, de sables, d'eau et même de solides [HL02]. En revanche la méthode est, à l'époque relativement coûteuse en temps de calcul et ne permet pas d'être intégrée à une application interactive.
L'objet de cet article est d'utiliser les « shaders », qui sont des programmes compilant et s'exécutant sur les cartes graphiques modernes, de profiter des parallélisations de calculs qu'ils offrent, pour réimplanter la méthode d'écran d'épingles dynamique décrite par Habibi et al. et la faire tourner en temps réel.
La section 2 rappelle brièvement la méthode d'écran d'épingles dynamique de Habibi et al. La section 3 fait un état de l'art des implantations de modèles physiques particulaires sur GPU. La section 4 présente l'implantation que nous avons réalisée. La section 5 relate l'incorporation de cette méthode dans une chaîne de simulation incluant les modèles particulaires amont simulés en temps réel. La section 6 décrit les performances obtenues. La section 7 propose une interface pour les utilisateurs. Finalement, la section 8 expose les résultats qui ouvrent eux-mêmes sur des applications nouvelles de l'écran d'épingles.

2 L'écran d'épingles dynamique

Pour plus de précision sur la méthode de l'écran d'épingles dynamique, on pourra se référer aux articles [HL02]. Il est cependant à noter que l'écran d'épingles est un réseau CORDIS masses-interactions comme décrit dans [LJF*91].

En entrée de l'écran d'épingles, il y a un ensemble de trajectoires de points que l'on nomme « marqueurs ». Ces marqueurs vont venir marquer l'écran, c'est-à-dire qu'ils vont agir dessus, créer une trace, sans que l'écran agisse sur eux en retour. Ainsi les trajectoires des marqueurs peuvent être fournies par n'importe quel processus (simulation, entrée par capteur) sans que celui-ci soit nécessairement opéré en temps réel synchrone avec la simulation de l'écran d'épingles. Les trajectoires peuvent donc être notamment fournies par une simulation amont hors-ligne d'un ensemble de particules dont la trace a été conservée dans un fichier.

L'écran d'épingles dynamique est un ensemble d'éléments physiques nommés « phyxels » (cf. figure 1). Ces phyxels, sortes d'épingles dynamiques, sont des algorithmes dont la sortie est une valeur scalaire qui correspond à la «hauteur » de l'épingle. Plus précisément un phyxel sera composé de deux masses, dont une fixe, en interaction unidimensionnelle. L'interaction la plus simple sera l'interaction linéaire visco-élastique, mais des interactions plus complexes seront envisagées dans cet article.
Les phyxels peuvent aussi interagir entre eux. Ainsi chaque masse mobile de chaque phyxel pourra être en interaction avec les masses des phyxels voisins. Ce faisant, on permet ainsi une diffusion spatiale de la trace opérée par les marqueurs. L'ensemble des travaux présentés dans cet article s'est restreint à l'utilisation d'un écran d'épingles 2D, c'est-à-dire à un tableau 2D de phyxels. Un phyxel possède donc au maximum 8 voisins dans cette configuration.
Les marqueurs eux-mêmes sont en interaction visco-élastique unidirectionnelle (l'épingle n'agit pas sur le marqueur), seuillée en distance, avec chaque masse mobile de chaque phyxel. C'est cette interaction qui permet aux marqueurs de laisser une trace sur l'écran.

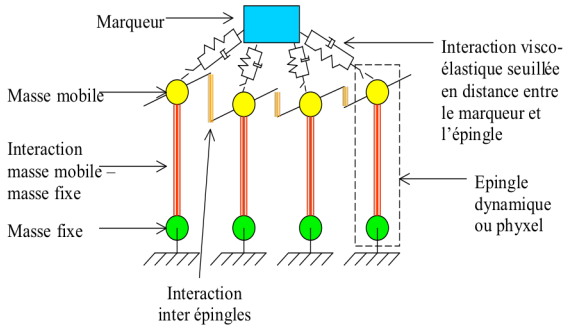

Figure 1 : *le modèle masses-interactions de l'écran d'épingles dynamique*

En sortie de l'écran d'épingles, on obtient bien un ensemble de valeurs scalaires correspondant aux hauteurs des épingles (distance masse fixe - masse mobile au sein d'un phyxel). Ce tableau de scalaire peut alors être utilisé en entrée d'une méthode de rendue. La méthode la plus simple proposée par [HL02], est le contrôle de la couleur d'un ou plusieurs pixels de l'écran par la combinaison d'une ou plusieurs hauteurs en sortie d'un ou plusieurs phyxel. On peut imaginer que les hauteurs contrôlent de la même manière des points de contrôle de surfaces dans un espace 3D ou encore des orientations de normales. Plusieurs méthodes de rendu seront présentées dans la suite de cet article et étendent les résultats présentés par [HL02].

3 Implantation de modèles physiques particulaires sur GPU

La programmation sur processeurs graphiques (GPU), non pas seulement pour des applications de rendu de scène 3D, mais pour tout type de calcul numérique, a connu un véritable succès ces dernières années. Ces types d'unités de calculs ont une architecture différente des processeurs présents sur les cartes mères (CPU). La différence fondamentale avec ces derniers est la parallélisation des calculs. Pour une revue de l'utilisation actuelle des GPU que peut en faire la communauté de l'informatique graphique, on pourra se reporter à [OLG*05].
Le principe général de programmation sur ce genre d'architecture est de stocker les variables de la simulation dans une structure de donnée propriétaire comme, par exemple, une texture qui était initialement conçue pour stocker les composantes RVBA des textures à afficher. Un programme appelé « shader » se compile et s'exécute sur une partie de la carte graphique. Il exécute la même tâche en parallèle sur les données stockées dans la structure de donnée en fonction de certains paramètres.
Plusieurs travaux récents se sont intéressés à l'implantation de systèmes de particules plongées dans des champs [KKK*05], en collision avec des objets rigides [KLR04] ou en interaction entre elles [GW05, LS06] sur des GPU en suivant ce principe de programmation. Nos travaux s'inscrivent dans cette direction, mais l'application est sensiblement différente.
Si les variables stockées dans les textures étaient des positions 3D de particules ou de points de contrôle de formes rendues directement, donc des valeurs vectorielles correspondant directement à un élément géométrique de la scène, les valeurs scalaires que nous produisons avec notre écran d'épingles dynamique ne peuvent correspondre directement à un élément de l'espace 3D et contrôlent donc un processus de rendu qui peut aller du contrôle direct de la couleur d'un pixel à une chaîne d'affichage plus complexe (en contrôlant, par exemple, les orientations des normales d'une surface servant à déterminer le rendu de leur éclairement). Les interactions implantées dans les articles cités étaient de simples interactions linéaires de type ressort-frottement. Par conséquent, une autre différence notable avec les travaux ayant fait intervenir des masses en interaction est que des interactions plus complexes sont susceptibles d'intervenir au sein du modèle de notre écran d'épingles dynamique comme le propose déjà Habibi dans sa thèse [H97].

4 Implantation de l'écran d'épingles dynamique sur GPU

4.1 Encapsulation des variables de la simulation

Les variables et certains paramètres de la simulation sont encapsulés dans des structures de données de type textures. Le GPU effectue ensuite des calculs sur ces textures.
Pour un écran d'épingles de m x n phyxels, une texture de taille n x m est créée sur la carte graphique (nous utilisons une texture rectangulaire pour avoir des coordonnées non normalisées). Chaque épingle est donc repérée par sa position en x et y sur la texture. Pour chaque épingle, la texture contient la position au temps t (composante Rouge), la position au temps t-1 (composante Verte). La composante bleue est conservée pour stocker certains paramètres des interactions amenés à évoluer.

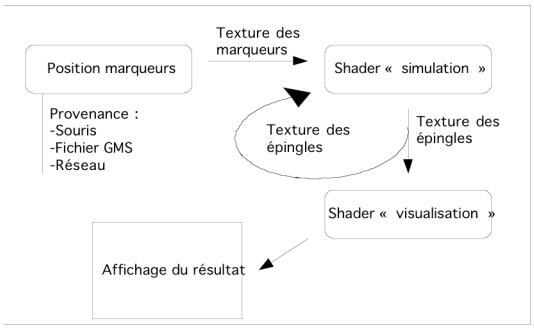

Figure 2 : *Schéma général de l'architecture logicielle de la simulation de l'écran d'épingles dynamique sur GPU*

4.2 Déroulement de l'algorithme

L'algorithme est une boucle composée de 3 passes distinctes (cf. figure 2):

Pour chaque boucle, il faut :

- Exécuter une passe « lecture » au cours de laquelle la position courante des marqueurs est inscrite dans une texture spécifique (texture « marqueur »).
- Lancer l'exécution de la passe «simulation» qui va inscrire dans la texture appelée texture « épingle » les nouvelles hauteurs des épingles en fonctions des différentes interactions.
- Effectuer une passe «rendu» qui va rendre à l'écran une image contrôlée par les valeurs scalaires de hauteur des épingles (sorties des phyxels).

4.2.1 La passe « Lecture »

Une texture est créée pour faire passer la position des marqueurs. Pour k marqueurs, la texture créée sera de taille k x 1, les deux premières composantes codant la position, les deux dernières étant conservées pour d'autres informations, comme une raideur ou un coefficient d'aplatissement [cf. HL02].
Dans notre implantation, les trajectoires des marqueurs peuvent provenir de 3 sources différentes :

- Un fichier de trajectoires produit éventuellement par une simulation hors-ligne amont. Le format utilisé est le format de codage de données de mouvement GMS [LEC*06].
- Un flot de données provenant d'un réseau. Une simulation amont peut alors être effectuée en temps-réel synchrone (voir section 5).
- La position d'un capteur comme la souris d'ordinateur.

Les trajectoires des marqueurs peuvent donc être parfaitement simulées par un modèle amont de type système de particules ou réseau masses-interactions. Cette simulation peut être temporellement décorrélée de la simulation de l'écran d'épingles dynamique puisque celui-ci n'agit pas sur les marqueurs. En revanche, si l'on souhaite que la simulation de l'écran d'épingles soit effectuée en temps réel, il convient que la lecture des trajectoires ne soit pas une phase bloquante. Ainsi la latence de l'accès matériel et de l'accès logiciel doit s'inscrire strictement dans une phase de la boucle de simulation.

4.2.2 La passe simulation

La passe «simulation » prend en entrée la position de tous les marqueurs, la hauteur de toutes les épingles au temps t et t-1 (hauteur de la masse mobile par rapport à la masse fixe), ainsi que tous les paramètres de l'écran d'épingles. Elle inscrit dans la texture en sortie (utilisation d'un frame buffer object), la nouvelle position des épingles après calcul, ainsi que leurs anciennes positions et les paramètres modifiables de l'interaction.

La simulation peut se décomposer elle-même en 4 phases :

- **Calcul de la force appliquée sur chaque épingle par les marqueurs**

Chaque marqueur interagit avec chaque épingle de son voisinage. L'interaction est une interaction visco-élastique seuillée en distance et unidirectionnelle [LJF*91].

- **Calcul de la force appliquée sur chaque épingle par son voisinage**

La masse mobile de chaque épingle est reliée par une interaction à certaines de ses voisines. Nous avons implanté des interactions avec (V4) les 4 voisines (les deux horizontales et les deux verticales) ainsi qu'avec (V8) les 8.

- **Calcul de la force appliquée sur chaque épingle par son interaction avec le sol**

La masse mobile de l'épingle est elle-même en interaction avec la masse fixe de la même épingle.

- **Détermination de la nouvelle hauteur des épingles**

La nouvelle hauteur de l'épingle est déterminée à partir de la sommation de toutes ces forces F_n

Ainsi, la nouvelle hauteur de l'épingle est donnée par :

$Z_{n+1} = 2.Z_n - Z_{n-1} + F_n(Tech^2/m)$

Avec m : la masse de l'épingle, Z_n et Z_{n-1} les hauteurs au temps t et t-1, et Tech la période d'échantillonnage du mouvement.

Ces calculs sont effectués par le shader « simulation » pour chaque épingle. Les valeurs trouvées et stockées (hauteur au temps t, t-1, longueur à vide) sont stockées dans la texture « épingles » et envoyées par la suite à la passe « rendu ».

4.2.3 La passe « rendu »

Nous effectuons une deuxième passe sur le GPU, pour permettre le rendu des scalaires (hauteurs) fournies par chaque phyxel (épingles). Son but est de passer du phyxel au pixel par un procédé plus ou moins complexe. Nous avons expérimenté 3 méthodes différentes de rendu de l'écran d'épingles dynamique:

Attribution de couleur simple :

La couleur est uniquement attribuée en fonction de la hauteur. On transforme 1 scalaire en 3 composantes (RVB). Cela permet une visualisation simple, mais assez esthétique. On peut nuancer la couleur par la hauteur de l'épingle correspondante. Par exemple, on peut définir le blanc pour une hauteur minimale, et le noir pour une hauteur maximale : on aura toutes les nuances des gris en visualisation.

Le Bump – Mapping :

Cette technique consiste à émuler un relief à partir d'une « normal map » et d'une position de source d'éclairage. Nous obtenons cet effet en créant une « normal map » par un « shader » en fonction de la dérivée de la hauteur des épingles (considération du voisinage). Il nous suffit alors d'éclaircir ou d'assombrir la lumière diffuse renvoyée par chaque pixel, pour obtenir un simulacre de relief. Cela permet une visualisation en fausse 3D mais ajoute un réel aspect de profondeur à l'observation.

Ajout d'une dimension :

Il peut être intéressant de visualiser l'écran d'épingles 2D que nous avons implanté dans un espace 3D. Ce type de rendu sera utilisé notamment pour le calibrage et pour une meilleure compréhension du phénomène dynamique.
Nous utilisons un « geometry shader » pour découper notre surface en un grand nombre de sommets, puis nous modifions la hauteur de ces sommets (ou vertex) en fonction de la hauteur de l'épingle lui correspondant. Nous pouvons grâce à cela, définir une surface variable en 3 dimensions réelles.

4.2.4 Initialisation

L'initialisation du système est effectuée en attribuant à la hauteur des épingles la longueur à vide de l'interaction avec la masse fixe. Cela permet de :

- s'assurer que toutes les épingles débutent à la même hauteur
- faire un premier test sur la stabilité de l'écran d'épingles (la hauteur des épingles ne varient pas avant une perturbation extérieur).
- préparer l'écran d'épingles à interagir avec les marqueurs. Cette initialisation est effectuée à chaque fois que l'utilisateur veut remettre l'écran d'épingles en position initiale (reset).

4.2.5 Codage des variables

La première implantation présentait des bruits dus à la quantification des données. En effet, la hauteur des épingles était codée sur 8 bits et réduite entre 0 et 1 (caractéristique d'openGL). Le bruit de quantification était alors très grand. Il convient donc de choisir une quantification bien supérieure pour résoudre ce problème.

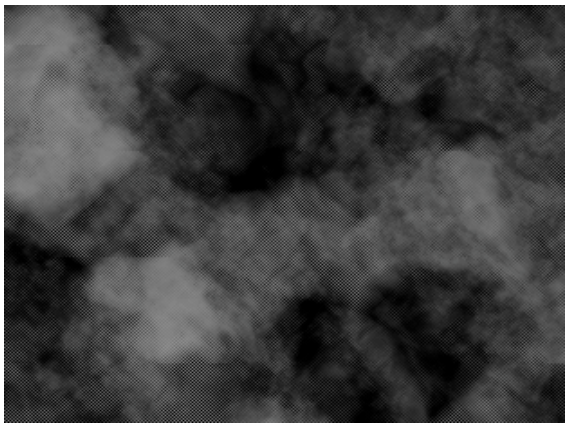

Figure 3 : *Comportement de l'écran d'épingles à l'équilibre pour un mauvais codage des Hauteurs (ici 8 bits)*

5 L'écran d'épingles dynamique dans une chaîne de simulation temps réel

Une simulation est effectuée sur une machine amont. Elle va, en sortie, délivrer la position de tous les marqueurs envoyés vers l'écran d'épingles. La question est de faire passer ces informations, le plus vite possible, de la machine amont à l'écran d'épingles implanté sur une autre machine.
Nous utilisons pour cela, le protocole UDP sur un réseau Ethernet. Une trame est constituée pour encapsuler les données et assurer la cohérence des informations à la réception. Cette trame est constituée de la sorte :

Fanion	Nombre de masse	Numéro Trame	N° masse	pos X	pos Y	N° masse	pos X	pos Y	…
FFFF.FFFF	3	245	0	12,5	3,3	1	-2,5	-3,8	…
32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	32 bits	

x Nombre masses

Elle permet à la machine réceptrice de se resynchroniser sur une nouvelle trame grâce au fanion. La lecture est donc non-bloquante, ce qui autorise le déroulement des deux simulations en parallèle. La communication est asynchrone dans le sens où seule une nouvelle trame va contraindre la simulation de l'écran d'épingles. Le nombre de masse permet de changer la simulation amont, et de faire passer l'information à l'écran d'épingles pour qu'il se réadapte au nouveau nombre de marqueurs. Le numéro de trame permet de repérer les frames sautées. Le numéro de masse est là pour assurer qu'il n'y ait pas de décalage majeur des indices. Si une trame est jugée incorrecte, l'utilisateur est prévenu et la frame correspondante est sautée.

6 Performances obtenues

La simulation de l'écran d'épingles dynamique a été réalisée sur un GPU GTX 8800 (768 Mo) branchée sur un PC avec CPU Dual Core II 2,4 GHz.
Les performances dépendent énormément du nombre de marqueurs. Nous les présentons ici pour une résolution de 1100 x 1100 épingles. Le tableau suivant a été obtenu dans le cas où les marqueurs proviennent d'une simulation amont différée (fichier GMS).

Nombre de marqueurs	1	5	10	20	50	100	300
Fréquence	693	510	358	223	105	56	20

Dans le cas de la chaîne temps réel (section5), il y a plusieurs limites à la vitesse du système :
-La vitesse à laquelle tourne la simulation amont
-La vitesse de la communication (dans notre cas, Ethernet à 100 Mbits/s, en protocole UDP)
-La vitesse de simulation de l'écran d'épingles

L'écran d'épingles étant destiné à la visualisation, nous pouvons tolérer une latence non visible, mais nous devons respecter une cadence cohérente avec la chaîne amont. Par rapport à nos performances, cela devient un problème pour un nombre de marqueurs importants (supérieur à 200) tant au niveau de la communication Ethernet que de la simulation de l'écran d'épingles.

Le tableau suivant présente les performances obtenues lors d'une simulation effectuée sur une autre machine envoyant les marqueurs à l'écran d'épingles (1100 x 1100 épingles) par le réseau. Pour respecter une certaine limite maximale dans la latence, des frames (une boucle de simulation) sont sautées en cas de surcharge.

Nombre de marqueurs	1	5	10	20	50	100	300
Fréquence	690	508	352	220	104	55	23

7 Interface

Nous avons porté aux usages notre écran d'épingles dynamique grâce à une interface graphique développée en QT4.

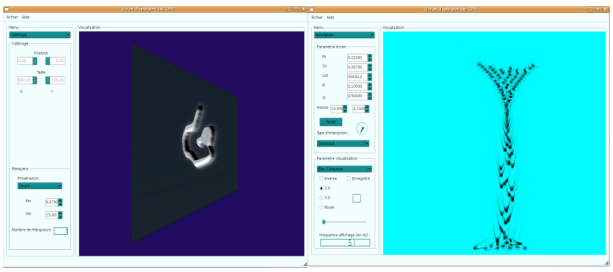

Figure 4 : *Interface utilisateur. A gauche : le mode calibrage. A droite : le mode simulation.*

L'interface se décompose en deux modes distincts qui modifient les options accessibles : un mode « calibrage » et un mode « simulation » (cf. figure 4).

7.1 Le mode calibrage

Il permet de paramétrer l'écran d'épingles avant la simulation, c'est-à-dire de régler sa taille et sa position en fonction des marqueurs. L'utilisateur pourra le redimensionner pour le centrer sur la partie qui l'intéresse. Dans ce mode, la simulation n'est pas en cours, l'écran d'épingles est représenté par une surface semi transparente et les marqueurs par des boules de diamètre de la taille de leurs rayons d'interactions. L'utilisateur a accès aux paramètres relatifs aux marqueurs (le seuil en distance de l'interaction, sa raideur, leur provenance...) ainsi qu'aux paramètres relatifs à la position et la taille de l'écran d'épingles dans l'espace.

7.2 Le mode simulation

Une fois les paramètres précédents réglés, l'utilisateur peut passer en mode simulation. Ce mode va lui permettre d'observer le comportement de l'écran d'épingles en temps réel. Il a ici accès à tous les paramètres des interactions, ainsi qu'un choix de plusieurs types d'interactions (ressort, viscosité, modification des longueurs à vide et d'autres...). De plus, l'interface permet de choisir parmi plusieurs types de rendu (ceux qui sont présents dans la section 8). Il peut ensuite jouer sur les couleurs et les textures. Une fonction d'enregistrement image par image de la scène et d'exportation au format TGA lui est aussi proposée. Un certain nombre de comportements globaux (interactions et paramètres donnés) sont préenregistrés, et l'utilisateur peut enregistrer et charger ses propres modèles à l'aide du menu. De plus l'utilisateur a le choix de visualiser la surface de l'écran d'épingles en 2D ou en 3D. Dans le cas de la 3D, il peut se déplacer et observer la surface (creusée, déformée par la hauteur des épingles). Dans le cadre du plaquage de texture sur une surface déformable, seule la sphère est actuellement proposée.

8 Résultats

8.1 Visualisation de modèles particulaires

Nous avons testé les capacités maximales en temps réel de notre application. Le modèle choisit, pour réaliser ce projet, est le même que celui décrit dans [HL02]. Il s'agit de visualiser un modèle masses-interactions de fumée proposé par [HLV*96]. Le modèle de la fumée est simulé sur une station amont (PC biprocesseur) en temps réel. La fumée est composée de 300 masses. Nous nous trouvons donc bien dans une situation de fonctionnement limite pour l'écran d'épingles qui tourne alors à une vingtaine de Hertz. La figure 5 montre les résultats obtenus :

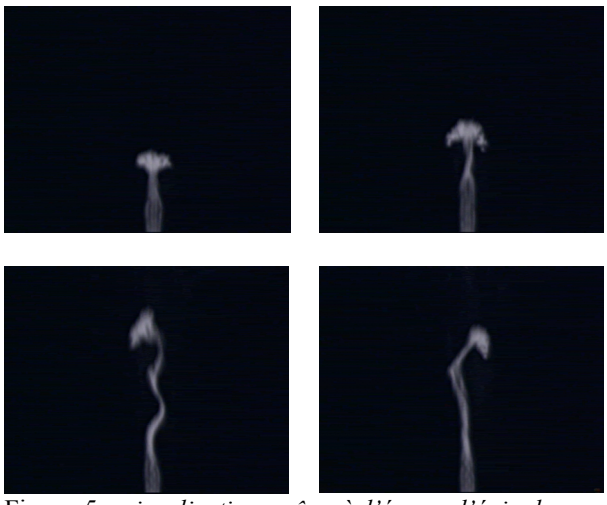

Figure 5 : *visualisation grâce à l'écran d'épingles dynamique d'un modèle masses-interactions de fumée à 300 masses*

8.2 Rendu « Bump mapping »

Dans les deux exemples suivants, l'écran d'épingles vient perturber les normales d'une surface pour créer, grâce au calcul de l'éclairage, un simulacre d'effet de relief.

8.2.1 Les sillons des bateaux

L'écran d'épingles est utilisé pour rendre les propagations sur la surface d'un lac (cf. figure 6). Il faut pour cela des paramètres qui permettent une propagation rapide (peu de viscosité et une forte raideur sur le voisinage). Les marqueurs sont placés au niveau de la ligne d'eau des bateaux (4 marqueurs par bateaux). La réfraction est simulée en appliquant une texture de fond. La réflexion est obtenue en effectuant une première passe projetant la scène à l'envers sur la surface de l'eau. La texture obtenue est ensuite stockée pour être envoyée au rendu final. Une « normal map » est générée en fonction de la hauteur des épingles. Elle permet d'aller lire à un autre endroit de la texture, pour simuler les déformations du reflet en fonction des vagues.

Figure 6 : *Vue de haut des sillons provoqués par 3 bateaux.*

8.2.2 Les traces des roues dans le sable

On dispose un marqueur sur chaque roue d'un véhicule. L'écran d'épingles va simuler le comportement d'un sol meuble sur lequel le véhicule se déplace. (cf figure 7). Pour cela, nous allons utiliser l'interaction plastique proposée par [CLH96]. Cette interaction présente un comportement d'hystérésis en modifiant elle-même sa longueur au repos lorsqu'elle dépasse un seuil d'élasticité, ce qui permet de créer des traces permanentes. Les épingles ne sont pas en interaction entre elles car aucun effet de propagation n'est désiré.

Figure 7 : *Véhicule sur sol meuble.*

8.3 Déformations de surface

Dans l'ensemble de ces exemples, l'écran d'épingles contrôle la déformation d'une surface. Cette surface n'est pas nécessairement un plan, nous avons par exemple, déformé une sphère comme le montre la figure 8 en contrôlant les rayons des sommets.

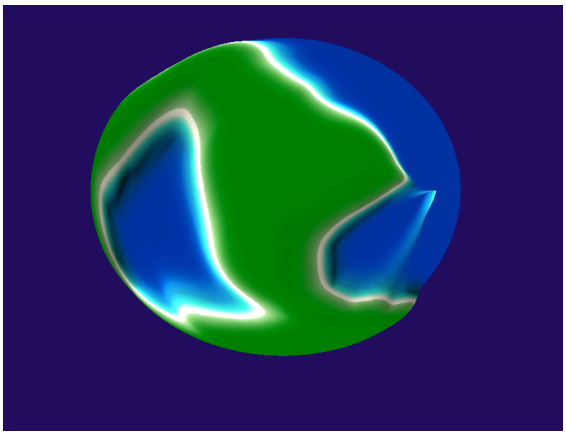

Figure 8 : *Une sphère déformée par l'écran d'épingles.*

En utilisant des interactions dont le comportement présente des irréversibilités, il est possible de créer des traces permanentes. On peut alors modéliser des phénomènes de morphogenèse comme la formation de profils de terrains (figure 9, 10) ou encore des fossilisations (figure 11,12).

Dans le cas des profils de terrains, l'utilisateur vient, à l'aide de la souris, « provoquer un

séisme» qui se propage. On obtient la configuration irréversible de l'écran d'épingles en supprimant toute élasticité entre la masse fixe et la masse mobile composant chaque épingle. Dans la figure 9, une carte d'altitude est appliquée à la surface déformée pour mieux rendre les différences de niveaux.

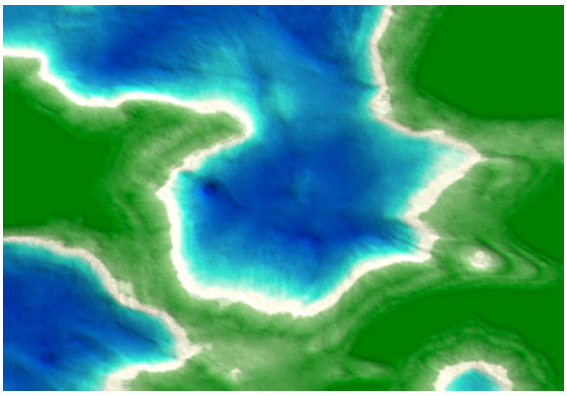

Figure 9 : *Génération de terrain par propagation.*

On peut aussi utiliser l'interaction plastique invoquée au paragraphe 8.2.2 pour modéliser des profils de terrains sablonneux (Figure 10). Dans ce cas, l'utilisateur vient lui-même « creuser » le sable :

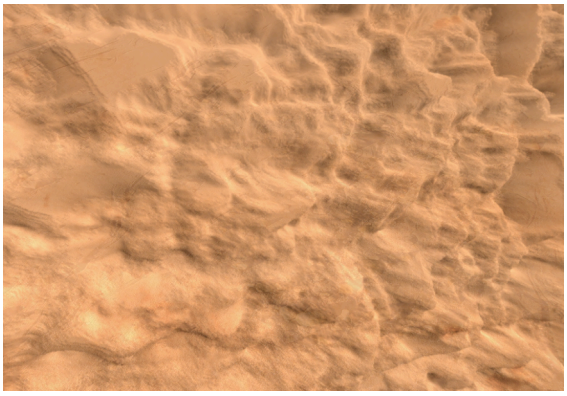

Figure 10 : *Profil de terrain sablonneux.*

Dans les figures 11 et 12, des sédimentations ont été réalisées par la même propagation que pour la génération de terrains, mais la propagation est beaucoup moins amortie (on joue sur une viscosité plus ou moins forte entre les épingles voisines). Une trace est ensuite effectuée par l'utilisateur dans le matériau résultant de cette sédimentation à l'aide de la souris. La permanence de la trace est rendu encore possible par une interaction hystérétique entre la masse fixe et la masse mobile de chaque épingle.

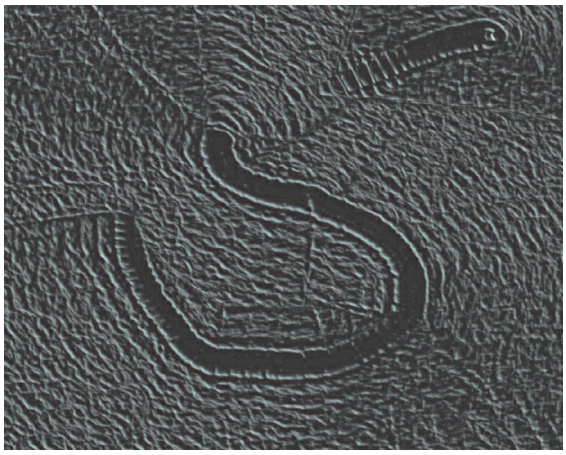

Figure 11 : *Fossilisation obtenue grâce à la trace laissée par l'utilisateur.*

.

Figure 12 : *Une autre trace fossile dans un matériau qui semble plus dur.*

9 Conclusion et perspectives

L'écran d'épingles peut être considéré comme une méthode d'extension spatiale de modèles particulaires, au même titre que les surfaces implicites. C'est en fait un générateur de surfaces implicites dynamiques puisqu'il génère un champ qui évolue dans le temps. Il peut aussi être vu comme un modèle physique en soit qui permet de modéliser un processus de gravure d'un genre nouveau.

Cet article a proposé une implantation temps réel de la méthode de l'écran d'épingles dynamique sur processeur graphique. On peut considérer que l'application arrive à ses limites pour un écran de 1100 par 1100 épingles et un nombre de marqueurs de l'ordre de 300.

L'implémentation a permis d'obtenir en temps réel des images qui ne pouvaient absolument pas l'être au moment où la méthode a été définie, vers 2002. Le temps réel a permis de l'incorporer à une interface interactive et

d'explorer de nouvelles possibilités notamment la modélisation en temps réel de profils de terrains ou de fossiles.
Plusieurs perspectives sont envisageables. L'une consiste à effectuer le travail théorique et l'implantation de la généralisation de l'écran d'épingles à un écran 3D (tableau 3D de phyxels). Il sera, de même, intéressant de connecter un système à retour d'effort à un certain niveau de la chaîne temps réel pour que l'utilisateur retrouve pleinement les sensations d'un graveur avec, en plus, une extension à des possibilités de comportements dynamiques nouveaux voire imaginaires qu'offre la méthode de l'écran d'épingles dynamique.

10 Remerciements

Nous tenons à remercier Mathias Paulin de l'Institut de Recherche en Informatique de Toulouse pour ses précieux conseils concernant la programmation sur cartes graphiques. Merci aussi à Julien Castet (ICA-INPG) et Jean-Loup Florens (ACROE) pour leur participation à la simulation temps réel de modèles de fumées, et le transfert des données de mouvement par réseau Ethernet.

11 Bibliographie

[AAA96] Hommage à Alexandre Alexeïeff et Claire Parker, édité par l'A.A.A (atelier d'animation d'Annecy), 1996.

[CL06] L. Chang, D. Liu. Deformable Object Simulation in Virtual Environment. Dans *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications* ISBN:1-59593-324-7, pp 327-330.

[CLH96] B. Chanclou, A. Luciani, A. Habibi. Physical Models of Loose Soils Dynamically Marked by a Moving Object. Dans *Proceedings of the Computer Animation*. Juin 1996, Genève, Suisse, pp 27-35.

[GW05] J. Georgii, R. Westermann. Mass-Spring Systems on the GPU. Dans *Simulation Practice and Theory 2005,* Elsevier Science Juillet 2005

[H97] A. Habibi. Visualisation d'objets très déformables. Relations Mouvement-Forme-Image. Thèse de l'INPG, Grenoble, 1997.

[HL02] A. Habibi, A. Luciani. Dynamic Particle Coating. Dans *IEEE Transactions on Visualisation and Computer Graphics*, vol. 8, no. 4, Octobre-Decembre 2002, pp 383-394.

[HLV*96] A. Habibi, A. Luciani, A. Vapillon. A Physically-Based Model for the Simulation of Reactive Turbulent Object. « Winter School on Computer Graphics », 1996, pp 113-122.

[KKK*05] J. Krüger, P. Kipfer, P. Kondratieva, R. Westermann. A Particle System for Interactive Visualisation of 3D Flows. Dans *IEEE transactions on visualization and computer graphics*, ISSN: 1077-26262005 Nov-Dec,11(6):744-56

[KLR04] A. Kolb, L. Latta, C. Rezk-Salama. Hardware-based Simulation and Collision Detection for large Particle Systems. Dans *Graphics Hardware 2004*. Août 2004, pp 123-132.

[LEC*06] A. Luciani, M. Evrard, N. Castagné, D. Couroussé, J.-L. Florens, C. Cadoz. A basic gesture and motion format for virtual reality multisensory applications. *Proceedings of the GRAPP conference*, February 2006.

[LJF*91] A. Luciani, S. Jimenez, J.-L. Florens, C. Cadoz, O. Raoult. Computational physics: a modeler simulator for animated physical objects. *Proceedings of the European Computer Graphics Conference and Exhibition*. Eurographics'91, pp 425-436, Vienna, Austria, Elsevier Ed, Sep. 1991.

[OLG*05] J.D. Owens, D. luebke, N. Govindaraju, M. Harris, J. Krüger, A. Lefohn, T. J. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. Dans *Proceedings of Eurographics 2005, State of the Art Reports.* Dublin, Irlande, 29 août - 2 septembre, pp 21-51.